[simula . research laboratory]

Work group meeting no. 2
-Parameters and value objects

INF5040 (Distributed systems)

Name: Sten L. Amundsen
Date: 7 September 2004

e-mail: stena@simula.no

[simula . research laboratory]

Agenda

- Parameter handling in servant operations.
- Attributes in servants.
- Transfer of value objects between client and servant.

simula . research laboratory

Parameters -IDL

- Parameters can be transported to/from operations.
- IDL supports in, out and inout.
- Java only supports in.
- Solution: IDL-to-Java use holders to wrap around out and inout parameters.

- IDL code

```
//signature for plain method/operation invocation
void printParameter(in string clientText);

//signatur with a return parameter
void getParameter(out string servantText);

//signatur with a return parameter
void multiplyNumber(inout long number);
```

simula . research laboratory

Parameters – Java

- IDL-to-Java compiler generates:
  - Stub and servant code (as normal) including Helper class.
  - Holder class for parameters.
  - Must implement usage of holder class in client and servant.

- Java code (xxOperations.java)

```
void printParameter(String clientText);

void getParameter(org.omg.CORBA.StringHolder servantText);

 void multiplyNumber(org.omg.CORBA.IntHolder number);
```

- Since predefined data types, example use CORBA's own Holder classes.

simula . research laboratory

Parameters – Java - Client

- Java client code (xxclient.java)

```
// send request with input parameter.
param.printParameter("The client says....");

//get parameter by using the Holder class.
org.omg.CORBA.StringHolder servantText = new
org.omg.CORBA.StringHolder();
param.getParameter(servantText);
System.out.println(servantText.value);

//inout parameter
org.omg.CORBA.IntHolder intValue = new org.omg.CORBA.IntHolder();
intValue.value = 2;
param.multiplyNumber(intValue);
System.out.println(intValue.value);
```

simula . research laboratory

Parameters – Java – Servant

- Java servant code (xxImpl.java)

in

out

inout

```
public void printParameter(String clientText){
        // Print parameter.
        System.out.println("Received from client:" +clientText);
}

public void getParameter(org.omg.CORBA.StringHolder servantText){
        String text = "Servant is happy.";
         // Text string into the CORBA StringHolder.
        servantText.value = text;
}

public void multiplyNumber(org.omg.CORBA.IntHolder number){
        //in parameter fetched from holder class.
       int holdInt = number.value;
        //out parameter is written to holder class.
       number.value = holdInt * 3;
}
```

simula . research laboratory

Attributes –IDL to Java

- In the IDL can define attributes in the servant.
- Attribute can be standard and defined types.

IDL code

```
attribute float price;
```

Java code (xxOperations.java)

```
float price(); //getMethod

void price(float val); //setMethod
```

simula . research laboratory

Attributes– Java – client and servant

- Java code (xxClient.java)

```
param.price(99999);//setMethod

float holdFloat = param.price();//getMethod
System.out.println(holdFloat);
```

- Java code (xxImpl.java)

```
public void price(float val) {
        this.priceForWork = val;
}

public float price() {
        return this.priceForWork;
}
```

simula . research laboratory

Value object -IDL

- Value objects can be single and in arrays
- Types in value objects can be enumeration constants

Enumeration constants

Value object

Array

```
 //define enumerated constants
enum DayInWeek{Monday, Tuesday, Wedensday, Thursday, Friday};

//defines a value object.
struct DayRate {
        DayInWeek day;
        float rate;
};

//signature with a return parameter
DayRate fillIn(in DayInWeek today, in float costperhour);

//define a one dimensional sequence (i.e. an array)
typedef sequence <DayRate> rateList;

//out using array (w/value objs) and boolean return value
boolean findLowestRate(out rateList list);
```

simula . research laboratory

Value object – Java

- IDL-to-Java compiler generates:
  - Object type for value object, array of value objects and enumeration constants.
  - Helper and holder classes for value object and enumeration constants.
- Java code (xxOperations.java)

```
//return value object
DayRate fillIn(DayInWeek today, float costperhour); //return value object


//Value object in array transferred as an out parameter.
boolean findLowestRate(hellodemo.ParamHandlerPackage.rateListHolder list);
```

simula . research laboratory

Value object – Java - Client

- Java client code (xxclient.java)

Enumeration constant

Value object

Array of Value objects

```
//Use value object as response.
DayRate holdData = param.fillIn(DayInWeek.Tuesday, 10);
System.out.println(holdData.day.value() +"  " +holdData.rate);

//Get data inside an array using the out parameter.
hellodemo.ParamHandlerPackage.rateListHolder list =
           new hellodemo.ParamHandlerPackage.rateListHolder();

boolean result = param.findLowestRate(list);

if (result) {
        DayRate[] hold = list.value;
         for (int i = 0; i < hold.length; i++) {
             System.out.println("  Day: " +hold[i].day);
             System.out.println("  Rate: " +hold[i].rate);
}
```

simula . research laboratory

HelloDemo example

- To understand the implementation, refer to the hellodemo example
- All students develop their own implementation using the provided IDL.

IDL file:

- /src/hellodemo.idl

Java source files:

- /src/hellodemo

Javadoc:

- /javadoc/hellodemo/index.html